# Chapitre 9

# Multimédia

# Introduction

- Android a une API multimédia capable de lire et d'enregistrer une large gamme de formats d'images, audio et vidéo aussi bien localement que via un flux.

- L'utilisation de cette API est basique. Vous allez utiliser le mécanisme des intents et des activités comme nous les avons décrits à plusieurs reprises dans les précédents chapitres.

- La lecture audio/vidéo peut se faire à travers plusieurs sources :

  - Une ressource locale dans le répertoire « raw » de votre application.

  - Un fichier stocké localement dans le système soit localement à l'application, ou bien dans une carte « SD ».

  - Un fichier en ligne, dans ce cas il y a un flux en lecture via le réseau.

- La classe à utiliser par votre application pour lire de l'audio et de la vidéo est : « MediaPlayer ».

- La classe à utiliser pour enregistrer de l'audio ou de la vidéo est : « MediaRecorder ».

- Les formats audio/vidéo supportés sont définis ici :

  http://developer.android.com/guide/appendix/media-formats.html

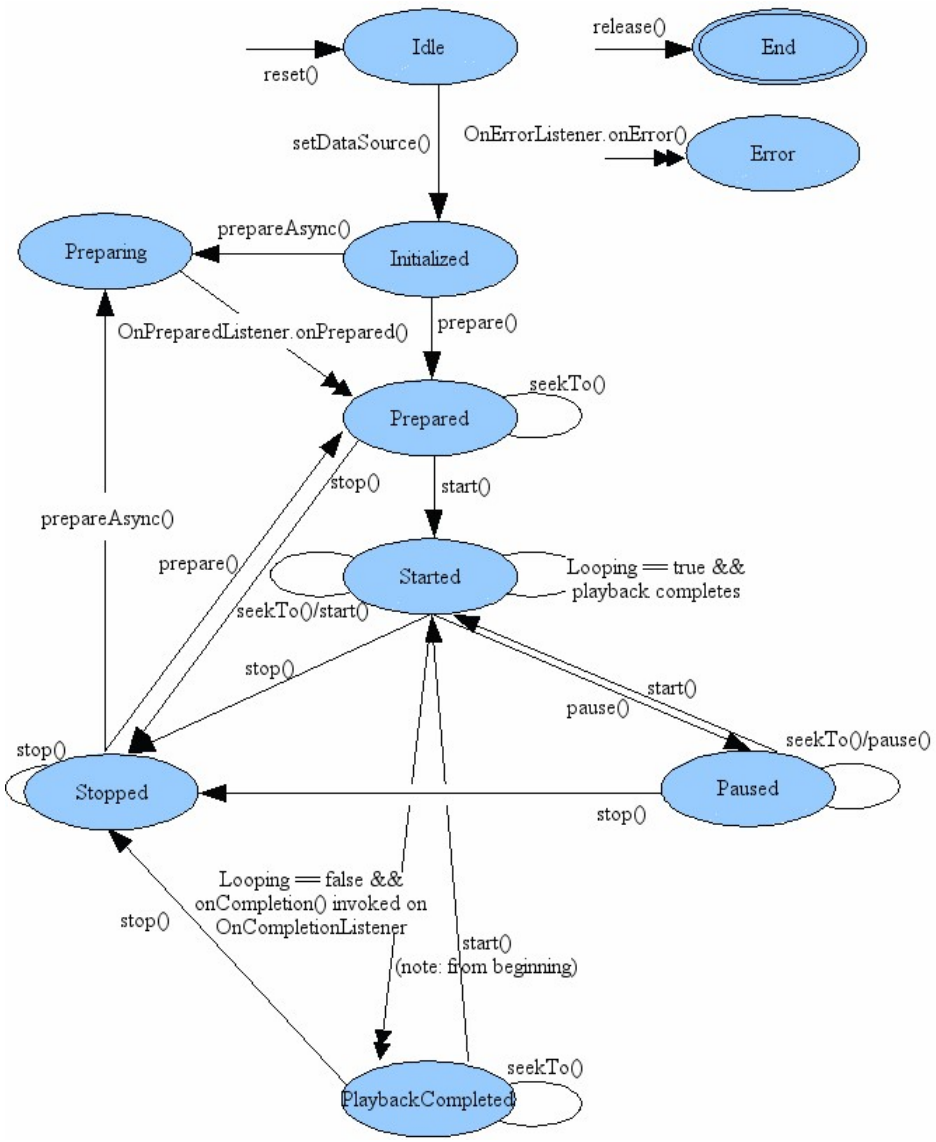
## Classe MediaPlayer

- Un objet de la classe « MediaPlayer » passe par plusieurs états.

- Le passage d'un état à un autre se fait à travers des opérations de contrôle.

- La transition d'un état à un autre peut se faire soit de manière synchrone (flèche simple sur le diagramme) ou bien de manière asynchrone (flèche double sur le diagramme).

  http://developer.android.com/reference/android/media/MediaPlayer.html


Idle
reset()
release()
End
setDataSource()
OnErrorListener.onError()
Error
Preparing
prepareAsync()
Initialized
OnPreparedListener.onPrepared()
prepare()
Prepared
seekTo()
stop()
start()
prepareAsync()
prepare()
Started
Looping == true &&
playback completes
seekTo()/start()
stop()
start()
pause()
seekTo()/pause()
stop()
Stopped
Paused
stop()
Looping == false &&
onCompletion() invoked on
OnCompletionListener
stop()
start()
(note: from beginning)
seekTo()
PlaybackCompleted

## Lecture audio

- À partir d'une ressource locale : déposez le fichier à lire dans le répertoire « res/raw » de l'application. Ce fichier sera ainsi référencé par la classe R. Nous pouvons donc faire appel à partir du code Java.

  Examiner l'exemple « C09: TestAudio ».

  - Créez une instance de « MediaPlayer » en utilisant pour cela la méthode « create » puis faites appel à la méthode « start » pour démarrer la lecture :

```
public void Lecture(View v){
    mp = MediaPlayer.create(TestAudioActivity.this,R.raw.essai);
    mp.start();
}
```

La méthode « create » fait implicitement appel à la méthode « prepare » pour initialiser le lecteur. La méthode « prepare » est en rapport avec l'état « Prepared ».

- Si le « MediaPlayer » n'est pas utilisé, que l'application a été arrêtée ou bien qu'elle a été détruite, il est important de détruire l'instance attachée au « MediaPlayer ». En effet, ce dernier est gourmand en ressources si vous ne voulez pas affamer votre appareil Android.

- Pour arrêter la lecture, vérifiez d'abord que l'instance de « MediaPlayer » est bien disponible. Dans le cas contraire, il n'est pas nécessaire d'intervenir.

```
if (mp != null) {
    mp.stop();
    mp.release();
    mp = null;
}
```

On arrête donc d'abord le lecteur, donc plus de son. Par la suite, on libère les ressources du lecteur et finalement, l'instance est remise à zéro.

- Si l'application est détruite, on s'assure aussi de libérer l'instance attachée au « MediaPlayer »

```
protected void onDestroy() {
    super.onDestroy();
    if (mp != null) {
        mp.release();
        mp = null;
    }
}
```

- À partir d'un fichier stocké localement :

  - On peut déclarer un intent en lui définissant une action sur un URI puis démarrer l'activité avec comme seul argument l'intent en question. Comme nous l'avons décrit plusieurs fois dans les précédents chapitres. Pour ce qui suit, nous allons utiliser une toute nouvelle approche propre à la classe « MediaPlayer ».

`Examiner l'exemple « C09 : TestAudioSD ».`

  - Nous allons lire le fichier « essai.mp3 » à partir de la carte « SD ». On commence par récupérer le chemin vers ce fichier. Par la suite, on déclare une instance de « MediaPlayer ». On lui fixe la cible. On prépare la ressource puis on commence à la lire.

  - Ne pas oublier d'autoriser votre application à accéder en lecture média audio : « android.permission.READ_MEDIA_AUDIO ». Cette permission est à ajouter depuis l'API 33.

```
public void Lecture(View v){
    String baseDir =
        Environment.getExternalStorageDirectory().getAbsolutePath();
    String fileName = "essai.mp3";
    String path = baseDir + "/Music/" + fileName;
    try {
        mp = new MediaPlayer();
        mp.setDataSource(path);
        mp.prepare();
        mp.start();
    }catch(Exception e) {
        Log.e(TAG, "error: " + e.getMessage(), e);
    }
}
```

- L'arrêt et la destruction de l'objet se font de la même manière que dans l'exemple précédent.

- À partir d'un flux internet :

`Examiner l'exemple « C09 : TestAudioStream ».`

- Nous aurions pu suivre exactement la même procédure que dans le cas d'une lecture d'un fichier déposé sur une carte « SD ». Cependant, dans ce cas, la lecture peut prendre un certain temps en fonction de l'état du réseau, la taille du fichier, etc.

- Il est donc préférable d'utiliser une approche asynchrone pour éviter un plantage « ANR ».

- Ainsi donc, au lieu d'utiliser la méthode « prepare » pour initialiser le lecteur, nous allons utiliser la méthode « prepareAsync ».

```
public void Lecture(View v){
    String path =
            "https://www.iro.umontreal.ca/~lokbani/essai.mp3";
    try {
        mp = new MediaPlayer();
        mp.setDataSource(path);
        mp.setOnPreparedListener(this);
        mp.prepareAsync();
        mp.setOnErrorListener(this);
    }catch(Exception e) {
        Log.e(TAG, "error: " + e.getMessage(), e);
    }
}
```

- La méthode « setOnPreparedListener » associe un écouteur d'événements dont la méthode « onPrepared » est appelée lorsque le lecteur est prêt.

- C’est cette méthode qui aura la tâche de démarrer la lecture du flux.

```
public void onPrepared(MediaPlayer mp)
    mp.start();
}
```

- En mode synchrone, les erreurs sont signalées par un code ou une exception. Tel n'est pas le cas en mode asynchrone. Il faut s'assurer de relayer l'information de l'erreur à l'application.

- On implémente l'écouteur « setOnErrorListener » dans notre application :

```
mp.setOnErrorListener(this);
```

- Par la suite, nous utilisons la méthode « onError » pour traiter cette erreur :

```
public boolean onError(MediaPlayer arg0, int arg1, int arg2) {
    return false;
}
```

L'argument « arg0 » est une instance du « MediaPlayer » dont l'erreur est associée. L'argument « arg1 » représente le type de l'erreur qui s'est produite. Finalement, l'argument « arg2 » est une information complémentaire sur l'erreur.

Si la méthode « onError » retourne « false », l'erreur n'est pas traitée. Dans ce cas de figure, et dans le cas où vous n'implémentez pas l'écouteur « setOnErrorListener », « onCompletionListener » est la méthode qui sera appelée.

- La méthode « onCompletionListener » associe un écouteur d'événements dont la méthode « onCompletion » sera appelée lorsque le média se termine.

- Il faudra ajouter la permission d'accès à l'internet dans le fichier « AndroidManifest » :

```
<uses-permission android:name="android.permission.INTERNET" />
```

## Enregistrement audio

- À noter que l'enregistrement audio à partir d'un émulateur est de piètre qualité. Il est vivement conseillé d'utiliser un « vrai » appareil Android pour effectuer les tests d'enregistrement.

`Examiner l'exemple « C09 : 27-MediaRecorder ».`

- On utilise cette fois-ci la classe « MediaRecorder ».

- On commence par déclarer une instance de cette classe :

```
myRecorder = new MediaRecorder(getApplicationContext());
```

- On définit les paramètres d'enregistrement :

  - D'abord la source, dans ce cas le micro :

    ```
    myRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
    ```

  - Le format du fichier, et l'encodeur audio à utiliser :

    ```
    myRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);

    myRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
    ```

    Il est recommandé d'utiliser le format « 3GP » pour une vidéo du type « H.263 » avec un encodeur audio « AMR ». Le format « MPEG-4 » peut confondre certains lecteurs.

- Par la suite, on informe l'enregistreur du nom du fichier à utiliser pour préserver l'enregistrement, puis on passe par les étapes de préparation puis le démarrage de l'enregistreur :

```
myRecorder.setOutputFile(this.mSampleFile.getAbsolutePath());
myRecorder.prepare();
myRecorder.start();
```

- L'arrêt de l'enregistrement se fait de la même manière que pour une lecture audio :

```
if (myRecorder != null){
    myRecorder.stop();
    myRecorder.release();
    myRecorder = null;
}
```

- Depuis l'API 33, il faut ajouter aussi la permission d'enregistrement audio : « android.permission.RECORD_AUDIO ».

## Lecture Vidéo

Examiner l'exemple « C09 : PlayingVideo ».

- On commence par ajouter une vue vidéo dans le fichier XML de l'activité :

```
<VideoView
android:id="@+id/videoPlayer"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center" />
```

- Ici, on se présente dans la même configuration que la lecture d'un flux par Internet. Comme on ne connaît pas les caractéristiques du fichier, il est préférable d'utiliser une approche asynchrone.

- On assigne une instance de « VideoView » au lecteur vidéo et on configure ses propriétés. Comme nous avons utilisé une approche asynchrone, la méthode « onPrepared » va se charger de démarrer la vidéo quand elle sera prête à la lecture.

```
videoPlayer = (VideoView) findViewById(R.id.videoPlayer);
videoPlayer.setOnPreparedListener(this);
videoPlayer.setOnCompletionListener(this);
videoPlayer.setKeepScreenOn(true);
videoPlayer.setVideoPath(root + "/Movies/" + pathToFile);
```

- Dans la méthode « onPrepared », nous avons ajouté la possibilité d'avancer la vidéo par un pas d'un cinquième sur la durée totale de la vidéo.

```
public void onPrepared(MediaPlayer vp) {
    if(videoPlayer.canSeekForward())
            videoPlayer.seekTo(videoPlayer.getDuration()/5);
    videoPlayer.start();
}
```

- Quand la lecture est terminée, on informe le système que l'activité a pris fin.

```
public void onCompletion(MediaPlayer  mp) {
    this.finish();
}
```

- Nous avons ajouté la possibilité à l'utilisateur de faire une pause en cliquant sur la souris. Pour relancer la lecture, on n'a qu'à cliquer de nouveau sur la souris.

```
public boolean onTouchEvent (MotionEvent ev){
    if(ev.getAction() == MotionEvent.ACTION_DOWN){
        if(videoPlayer.isPlaying()){
            videoPlayer.pause();
        } else {
            videoPlayer.start();
        }
        return true;
    } else {
        return false;
    }
}
```

- Ne pas oublier d'autoriser votre application à accéder en lecture média vidéo : « android.permission.READ_MEDIA_VIDEO ». Cette permission est à ajouter depuis l'API 33.

## Prendre une photo

- Android permet de capturer des images et des vidéos à travers l'API « Camera » ou bien via l'intent « Camera ».

- La classe « Camera » permet de contrôler le dispositif de caméra. Elle permet à votre application de prendre des photos ou des vidéos.

- L'intent associé à la prise d'une photo est :
  « `MediaStore.ACTION_IMAGE_CAPTURE` ».

- Ajoutez dans le fichier « Android Manifest » l'utilisation de la caméra :

```
<uses-feature android:name="android.hardware.camera"  android:required="true"/>
```

- Si vous devez sauvegarder l'image sur un média externe, n'oubliez pas d'ajouter :

```
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

- Si votre caméra a par exemple l'autofocus, n'oubliez pas d'activer cette propriété dans le fichier manifeste de l'application :

```
<uses-feature android:name="android.hardware.camera.autofocus" />
```

- L'option la plus simple pour prendre une photo est de passer par les intents.

  Examiner l'exemple « C09 : imagepicker ».

- Il faut procéder comme suit :

  - Créez un intent pour capturer une image :

```
Intent pictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
```

  - Ajoutez des informations complémentaires à cet intent, par exemple, le nom du fichier à utiliser pour sauvegarder l'image :

    Pour une sauvegarde directe dans le « MediaStore » :

```
pictureIntent.putExtra(
        MediaStore.EXTRA_OUTPUT,
        MediaStore.Images.Media.EXTERNAL_CONTENT_URI.toString()
    );
```

Ou bien via un fichier à déposer quelque part (la carte « SD » par exemple ») :

```
File f;

pictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(f));
```

Ou bien via un fichier déposé ailleurs dans votre appareil en utilisant un fournisseur de contenu :

```
File photoFile;
photoFile = createImageFile();
Uri photoUri = FileProvider.getUriForFile(this, getPackageName()
                                          +".provider", photoFile);
pictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoUri);
```

Le fournisseur est défini dans le fichier « AndroidManifest.xml ». La ressource associée est définie dans le fichier « file_paths.xml » dans le répertoire « res/xml ».

- Démarrez l'intent associé à la caméra avec l'appel :

```
someActivityResultLauncher.launch(pictureIntent);
```

- Le résultat est récupéré via la méthode :

```
public void onActivityResult(ActivityResult result)
```

L'argument va permettre de valider la prise de photo ou l'annulation de celle-ci.

Par la suite, on récupère l'image de l'intent et on l'affiche dans le « ImageView » qui a été déclaré dans le fichier XML de l'activité :

```
private String imageFilePath = "";
File image = File.createTempFile(imageFileName, ".jpg", storageDir);
imageFilePath = image.getAbsolutePath();
imageView.setImageURI(Uri.parse(imageFilePath));
```

- On peut utiliser aussi l'API « Camera ». Examinez pour cela l'exemple disponible sur ce lien :

  http://www.vogella.com/articles/AndroidCamera/article.html

## Enregistrer une vidéo

La capture vidéo se fait de la même manière que la prise d'une photo. La différence se situe dans l'utilisation de l'intent en rapport avec la vidéo.

- Créez un intent pour capturer une vidéo :

```
Intent videoIntent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
```

- Ajoutez des informations complémentaires à cet intent, par exemple, le nom du fichier à utiliser pour sauvegarder l'image :

  Pour une sauvegarde directe dans le « MediaStore » :

```
videoIntent.putExtra(
        MediaStore.EXTRA_OUTPUT,
        MediaStore.Video.Media.EXTERNAL_CONTENT_URI.toString()
    );
```

Ou bien via un fichier à déposer quelque part (la carte « SD » par exemple ») :

```
File f;

videoIntent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(f));
```

Ou bien via un fichier déposé ailleurs dans votre appareil en utilisant un fournisseur de contenu :

```
File videoFile;
videoFile = createVideoFile();
Uri videoUri = FileProvider.getUriForFile(this, getPackageName()
                                          +".provider", videoFile);
videoIntent.putExtra(MediaStore.EXTRA_OUTPUT, videoUri);
```

Le fournisseur est défini dans le fichier « AndroidManifest.xml ». La ressource associée est définie dans le fichier « file_paths.xml » dans le répertoire « res/xml ».

- Démarrez l'intent associé à la caméra avec l'appel :

```
someActivityResultLauncher.launch(videoIntent);
```

- Le résultat est récupéré via la méthode :

```
public void onActivityResult(ActivityResult result)
```

L'argument va permettre de valider la prise de photo ou l'annulation de celle-ci.

Par la suite, on récupère l'image de l'intent et on l'affiche dans le « VideoView » qui a été déclaré dans le fichier XML de l'activité :

```
private String videoFilePath = "";
File vidéo = File.createTempFile(videoFileName, ".mp4", storageDir);
videoFilePath = video.getAbsolutePath();
videoView.setImageURI(Uri.parse(videoFilePath));
```

Examiner l'exemple « C09 : CaptureVideo ».

- Ajoutez dans le fichier « Android Manifest » l'utilisation de la caméra :

```
<uses-feature android:name="android.hardware.camera"  android:required="true"/>
```

- On peut utiliser aussi l'API « Camera ». Examinez pour cela l'exemple disponible sur ce lien :

  http://developer.android.com/guide/topics/media/camera.html

- Google suggère d'utiliser la nouvelle API, voici quelques exemples (kotlin!) :

  https://github.com/android/camera-samples

## Synthèse vocale

Examiner l'exemple « C09 : txt2spk ».

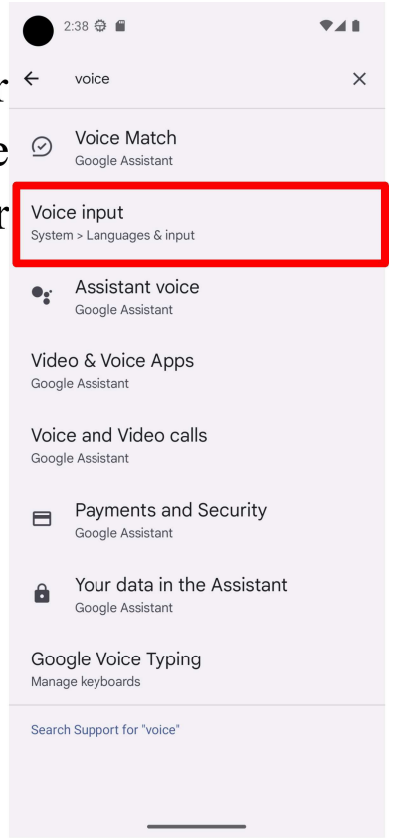
- On vérifie que l'appareil a un TTS et qu'il est possible de l'utiliser :

```
Intent checkIntent = new Intent();
checkIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
startActivityForResult(checkIntent, REQ_TTS_STATUS_CHECK);
```

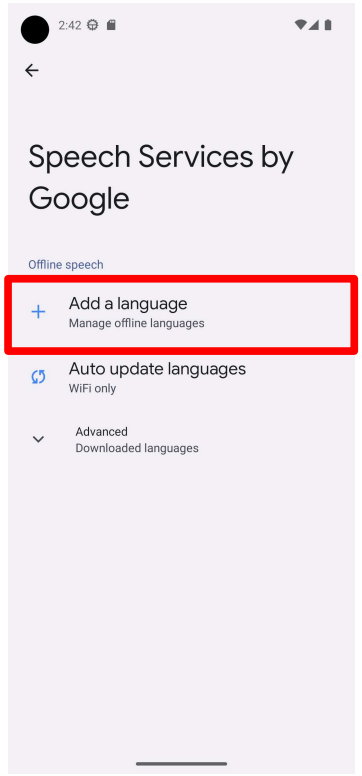
```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == REQ_TTS_STATUS_CHECK) {
            switch (resultCode) {
            case TextToSpeech.Engine.CHECK_VOICE_DATA_PASS:
                // TTS is up and running
                mTts = new TextToSpeech(this, this);
                break;
            case TextToSpeech.Engine.CHECK_VOICE_DATA_FAIL:
                // missing data, install it
                Intent installIntent = new Intent();
                installIntent.setAction(
                    TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA);
                startActivity(installIntent);
                break;
            default:
                Log.e(TAG, "Got a failure. TTS apparently not available");
            }
        }
        else {
            // Got something else
        }
    }
```

- Depuis la version « Jelly Bean », « API 16 », il est possible d'avoir localement sur l'appareil d'autres langues pour la synthèse vocale.

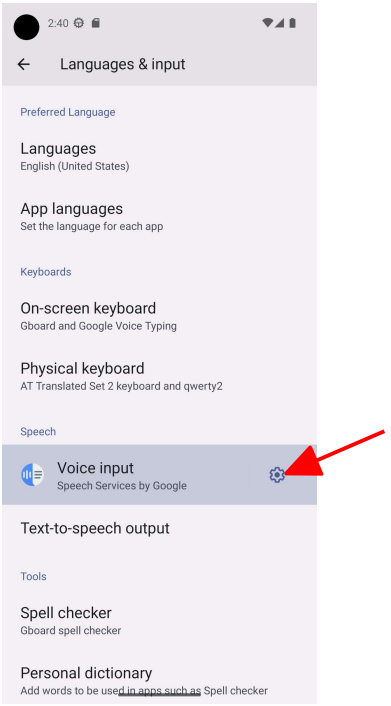
- Suivre la procédure décrite dans le cas de votre appareil. Sur un émulateur (API 33), chercher dans le panneau de configuration « Voice ». Puis cliquer sur « Voice input » pour télécharger des voix pou un usage local.



Cliquer sur le bouton de configuration de « Voice input »




Puis « add language ».



- Par défaut, la langue du synthétiseur est celle de l'appareil. Ainsi si l'appareil est en anglais, il va choisir la langue anglaise pour la voix à synthétiser. Pour fixer la langue dans le programme, indépendamment de celle de l'appareil :

```
mTts.setLanguage(Locale.FRENCH);
```

## Reconnaissance vocale

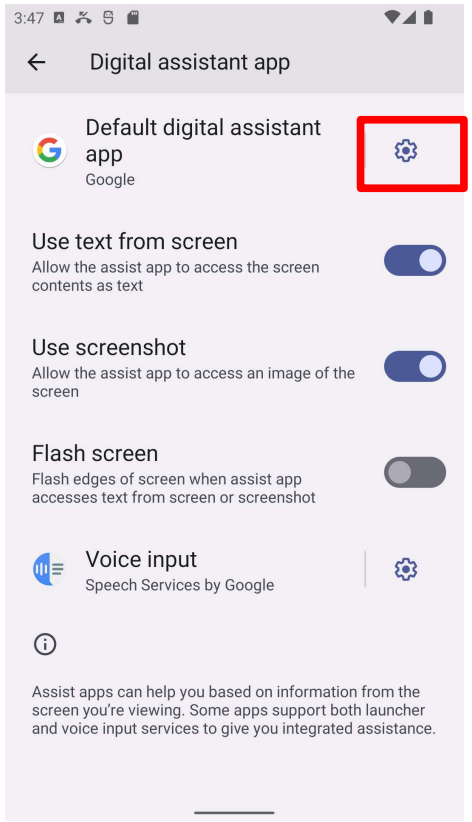
Examiner l'exemple « C09 : VoiceRecognitionDemo ».

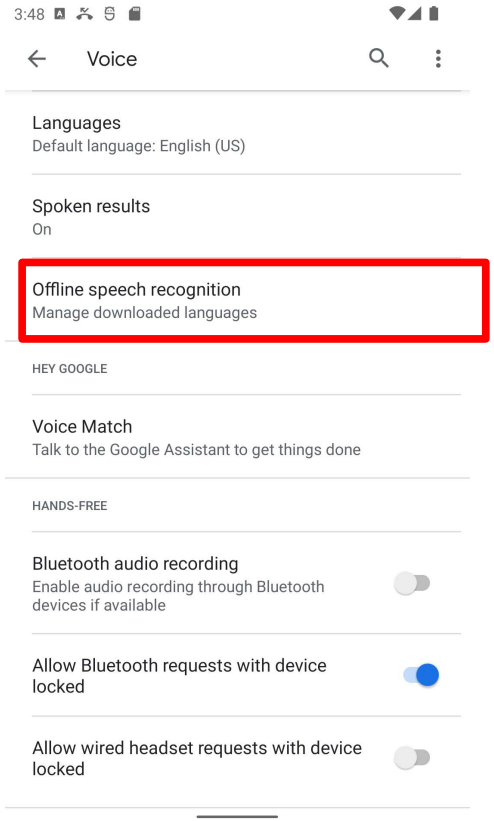
- La reconnaissance vocale va avoir lieu via l'internet. Il faudra donc fournir les permissions :

```
<uses-permission android:name="android.permission.INTERNET" />
```

- Le système de reconnaissance communique avec les serveurs de Google pour produire un résultat. La permission « Internet » n'est pas plus nécessaire depuis la version « Jelly Bean », « API 16 ». Il est, en effet, possible d'installer localement sur l'appareil, les modèles nécessaires à l'engin de reconnaissance.

- Suivre la procédure décrite dans le cas de votre appareil. Sur un émulateur (API 32), chercher dans le panneau de configuration « Digital Assistant app » et cliquer sur sa configuration. Puis cliquer sur « Offline speech recognition » pour télécharger les langues pour un usage local.



- On va utiliser l'intent « `RecognizerIntent.ACTION_RECOGNIZE_SPEECH` » :

```
Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
intent.putExtra(RecognizerIntent.EXTRA_PROMPT,
                                    "AndroidBite Voice Recognition...");
startActivityForResult(intent, REQUEST_CODE);
```

- Le résultat obtenu est sous la forme d'une liste des N meilleures propositions. Cette liste sera stockée dans un tableau pour être affichée par la suite à l'utilisateur de l'application :

```
if (requestCode == REQUEST_CODE && resultCode == RESULT_OK){
    ArrayList<String> matches = data.getStringArrayListExtra(
            RecognizerIntent.EXTRA_RESULTS);
    resultList.setAdapter(new ArrayAdapter<String>(this,
             android.R.layout.simple_list_item_1,matches));
}
```

- Par défaut, la langue du système de reconnaissance est celle de l'appareil. Ainsi si l'appareil est en anglais, il va choisir la langue anglaise pour la voix à reconnaître. Pour fixer la langue dans le programme, indépendamment de celle de l'appareil :

```
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE,
                                        Locale.FRENCH.toString());
```

## Bibliographies

Chapitre 27, « Android Multimedia », Victor Matos (lien non actif, web.archive.org)
http://marek.piasecki.staff.iiar.pwr.wroc.pl/dydaktyka/mc_2012/readings/Android-Chapter27-Multimedia%20%28M%29.pdf

Media and Camera (et les exemples associés)
https://developer.android.com/guide/topics/media/index.html

La chanson (MP3) utilisée dans les exemples audio :
http://archive.org/details/revoid_2004_04_04.flac16
Morceau : Obscure Terrain

Exemple « PlayingVideo »
http://eagle.phys.utk.edu/guidry/android/playingVideo.html

La vidéo utilisée dans l'exemple :
http://camendesign.com/code/video_for_everybody/test.html

Codelab pour l'utilisation de CameraX :

https://developer.android.com/training/camerax

Exemple « txt2spk »
http://www.cs.uwyo.edu/~seker/courses/4730/example/and-txt2spk.zip

Exemple « VoiceRecognitionDemo »
http://androidbite.blogspot.ca/2013/04/android-voice-recognition-example.html